Distributed Maximum A Posteriori Estimation for Multi-robot Cooperative Localization

Esha Nerurkar, Stergios Roumeliotis, Agostino Martinelli

Distributed Maximum A Posteriori Estimation for Multi-robot Cooperative Localization

Esha D. Nerurkar, Stergios I. Roumeliotis, and Agostino Martinelli

***Abstract*— This paper presents a *distributed* Maximum A Posteriori (MAP) estimator for multi-robot Cooperative Localization (CL). As opposed to centralized MAP-based CL, the proposed algorithm reduces memory requirements and computational complexity by distributing data and computations amongst the robots. Specifically, a *distributed data-allocation scheme* is presented that enables robots to simultaneously process and update their local data. Additionally, a *distributed Conjugate Gradient* algorithm is employed that reduces the cost of computing the MAP estimates while utilizing all available resources in the team, and increasing robustness to single-point failures. Finally, a computationally efficient *distributed marginalization* of past robot poses is introduced for limiting the size of the optimization problem. The communication and computational complexity of the proposed algorithm is described in detail, while extensive simulations studies are presented for validating the performance of the distributed MAP estimator and comparing its accuracy to that of existing approaches.**

I. INTRODUCTION

Autonomous mobile robot teams have the potential to be used for space and underwater exploration [1], surveillance [2], and search and rescue missions [3]. Accurate localization (i.e., estimating the position and orientation) of these robots is a prerequisite for the successful execution of higher-level tasks. GPS measurements, which ensure bounded uncertainty in the robots' pose estimates, are often unreliable (e.g., in urban canyons) or unavailable (e.g., in space, underwater, caves, etc). An alternative approach for multi-robot applications is Cooperative Localization (CL), where groups of communicating robots use relative measurements (such as distance, bearing and orientation) to *jointly* estimate their poses, resulting in increased accuracy for the entire team.

Recently, estimation algorithms such as the Extended Kalman Filter (EKF) [4], Maximum Likelihood Estimation (MLE) [5], and Particle Filters [6], have been used to solve the CL problem. In most cases, however, these algorithms require that all robot measurements are communicated to a fusion center (FC), which makes them susceptible to single-point failures. Additionally, the communication and computation cost, for large teams of robots, becomes prohibitive for real-time implementations. Moreover, distributed versions of these approaches are often based on approximations that provide no guarantees for their convergence (cf. Section II).

This work was supported by the University of Minnesota (DTC), and the National Science Foundation (EIA-0324864, IIS-0643680, IIS-0811946)

E. D. Nerurkar, and S. I. Roumeliotis are with the Department of Computer Science, University of Minnesota, USA {nerurkar,stergios}@cs.umn.edu

A. Martinelli is with INRIA Rhone Alpes, Grenoble, France agostino.martinelli@inrialpes.fr

In this paper, we introduce a *distributed* MAP-based CL algorithm that, in contrast to centralized approaches, harnesses the computational and storage resources of *all* robots in the team to reduce the computational complexity and achieve real-time performance. Since the robots' motion and measurement models are *non-linear*, the MAP estimator improves the accuracy of their pose estimates over the *entire* trajectory by acting as a *smoother* and reducing linearization errors. Specifically, MAP-based CL is formulated as a non-linear least-squares (LS) problem (Section III) and solved iteratively using the Levenberg-Marquardt (LM) minimization algorithm (Section IV).

The distributed MAP-based CL algorithm's storage, computation, and communication efficiency stems from: (i) The *distributed data-storage scheme* which allows for parallel processing of information locally available to each robot (Section V-A). (ii) The *Distributed Conjugate Gradient* (DCG) algorithm employed at each iteration of the LM minimization process with cost *at most quadratic* in the number of robots (Section V-B). (iii) The *distributed marginalization* of past robots' poses that limits the size of the optimization problem and whose computational complexity is *quadratic* in the number of robots (Section V-C).

II. RELATED WORK

In this section, we briefly review centralized and distributed algorithms for CL when *no map* of the environment is available to the robots (as is the case in e.g., [6]).

A. Centralized Cooperative Localization

Early work on CL considered robots operating as "portable beacons" [7], [8], [9]. Specifically, the robot team is divided into two sub-teams, one of which is moving while the other remains stationary acting as beacons. This process is alternated till all the robots reach their final destination. The main drawback of this approach is that it constraints the motion of the robots. Additionally, no information is provided about the computational and communication complexity of CL.

An EKF-based algorithm for CL was introduced in [10]. This approach allows the robots to propagate their state and covariance estimates independently by decomposing the centralized EKF-based CL into N communicating filters [4]. However, during each update step, all robots need to communicate with each other and update the covariance matrix for all pose estimates. This induces a computation cost of $O(N^2)$, where N is the number of robots in the team, for processing each relative position measurement. Considering that the total number of robot-to-robot measurements per

Preprint submitted to 2009 IEEE International Conference on Robotics and Automation. Received September 15, 2008.

time step can be as high as $N(N-1)$, the overall processing cost becomes $O(N^4)$. Even if the computations are equally distributed among the N robots, the cost is still prohibitively high ($O(N^3)$) for real-time operation of large teams.

A centralized MLE-based approach to CL is presented in [5], where the resulting non-linear optimization problem is solved directly using the CG algorithm with a cost of $O(K^2N^3)$, where K is the number of time steps considered. Similar in spirit is the centralized MAP-based CL algorithm of [11] which employs a sparse QR solver at each iteration of the LM method used for solving the non-linear minimization problem. The main drawback of both previous approaches is that all computations are performed centrally, rendering them susceptible to single-point failures. Additionally, application of these methods is limited to small robot teams due to their high processing cost.

B. Decentralized Cooperative Localization

In order to reduce the computational complexity of CL, various decentralized sub-optimal EKF-based algorithms have been proposed. Approximations that do not require uninterrupted inter-robot communication are presented in [12], where the Interlaced Kalman filter [13] is applied, and in [14] where state-estimates exchange is employed. Martinelli [15] proposes an approach based on a hierarchy of EKFs. In this case, the robot group is divided into sub-teams. The states of the robots in a sub-team are estimated by its leader using an EKF. Depending on the number of leaders, the leaders themselves can also form sub-teams and the same division of processing is repeated in a hierarchical manner ensuring that the size of each sub-team is bounded. The main drawback of these approaches is that in order to reduce the computational complexity of EKF-based CL some (or even all in the case of [12]) correlations are ignored, which may lead to overly optimistic and inconsistent estimates.

A decentralized version of the ML-based CL algorithm of [5] is presented in [16]. In this case, the non-linear optimization problem is divided into N sub-problems, one for each robot. In this approximation, every robot independently minimizes the part of the cost function that contains terms corresponding to: (i) its proprioceptive (odometry) measurements and (ii) exteroceptive (robot-to-robot relative pose) measurements involving the robot. During this process, the pose estimates of the other robots are considered constant. All robots periodically broadcast their updated pose estimates and the same process is repeated. The main drawback of this algorithm is that there exists no proof that it will converge even to a local minimum. Additionally, the authors provide no information about the processing requirements of their approach.

C. Proposed approach

In this paper, multi-robot CL is formulated as a MAP estimation problem. Its solution is found by employing the LM non-linear minimization algorithm that *guarantees fast convergence* to at least a local minimum.[1] During each iteration of LM, the resulting linearized system of equations is solved *in parallel* by all robots using the distributed CG (DCG) algorithm. This in effect, reduces the computational complexity of CL by a factor of N. A key advantage of DCG (iterative) over direct algorithms (e.g., distributed Gauss-Elimination) is that it provides an *intermediate solution* at every iteration. Furthermore, contrary to other iterative methods (such as the Jacobi algorithm) that converge only asymptotically, DCG converges within a *bounded number of iterations*. These key features of DCG allow the robots to trade processing for accuracy when computing resources are scarce (e.g., during time-critical tasks). This approach to CL along with the *distributed marginalization* of past robot poses, enables the team to perform real-time CL using limited computation and communication resources.

III. Problem Formulation

Consider a team of N communicating robots navigating in 2D while performing CL. In this case, the state vector $\mathbf{x}_k = [\mathbf{x}_k^{1\,T}, \mathbf{x}_k^{2\,T}, \ldots, \mathbf{x}_k^{N\,T}]^T$, $i = 1, \ldots, N$, where $\mathbf{x}_k^i = [x_k^i, y_k^i, \phi_k^i]^T$, contains the position and orientation of all robots at time-step k. Each robot carries proprioceptive (odometric) sensors that provide linear, $v_{m_k}^i$, and rotational, $\omega_{m_k}^i$, velocity measurements. The motion model for robot i is given by

$$\mathbf{x}_k^i = \mathbf{f}(\mathbf{x}_{k-1}^i, \mathbf{u}_{k-1}^i, \mathbf{w}_{k-1}^i) \tag{1}$$

where $\mathbf{f}$ is in general a non-linear function, and $\mathbf{u}_{k-1}^i = [v_{m_{k-1}}^i, \omega_{m_{k-1}}^i]^T$. The noise in the linear and rotational velocity measurements is represented by $\mathbf{w}_{k-1}^i = [w_{v_{k-1}}^i, w_{\omega_{k-1}}^i]^T$, assumed to be additive zero-mean white Gaussian with covariance $\mathbf{Q}_{k-1}^i$.

Additionally, all robots carry exteroceptive sensors that allow them to uniquely identify other robots in the team and measure their relative distance and bearing. The measurement model for robot i measuring robot j is

$$\mathbf{z}_k^{i,j} = \mathbf{h}(\mathbf{x}_k^i, \mathbf{x}_k^j) + \mathbf{n}_k^{i,j} \tag{2}$$

with $\mathbf{h} = [d_k^{i,j}, \theta_k^{i,j}]^T$, where $d_k^{i,j}$ and $\theta_k^{i,j}$ are the true distance and bearing respectively, from robot i to robot j at time-step k and $\mathbf{n}_k^{i,j} = [n_{d_k}^{i,j}, n_{\theta_k}^{i,j}]^T$ is the additive zero-mean white Gaussian measurement noise with covariance $\mathbf{R}_k^{i,j}$.

Our objective is to find the MAP estimate of the robots' poses, $\mathbf{x}_{0:K-1}$, up to time-step $K-1$ given the measurements $\mathbf{z}_{0:K-1}$ and $\mathbf{u}_{0:K-2}$. The MAP-estimator is formulated as

$$\begin{aligned}
\hat{\mathbf{x}}_{0:K-1} &= \arg\max p(\mathbf{x}_{0:K-1} | \mathbf{z}_{0:K-1}, \mathbf{u}_{0:K-2}) \\
&= \arg\max \frac{1}{p(\mathbf{z}_{0:K-1})} p(\mathbf{z}_{0:K-1} | \mathbf{x}_{0:K-1}) p(\mathbf{x}_{0:K-1} | \mathbf{u}_{0:K-2}) \\
&= \arg\max \frac{1}{p(\mathbf{z}_{0:K-1})} \prod_{k=0}^{K-1} p(\mathbf{z}_k | \mathbf{x}_k) \prod_{k=0}^{K-2} p(\mathbf{x}_{k+1} | \mathbf{x}_k, \mathbf{u}_k) \cdot p(\mathbf{x}_0) \\
&= \arg\min \left(- \sum_{k=0}^{K-1} \log p(\mathbf{z}_k | \mathbf{x}_k) - \sum_{k=0}^{K-2} \log p(\mathbf{x}_{k+1} | \mathbf{x}_k, \mathbf{u}_k) - \log p(\mathbf{x}_0) \right)
\end{aligned} \tag{3}$$

[1] Note that LM is optimal up to linearization errors. However, as is the case for all non-linear minimization algorithms, convergence to the global minimum is guaranteed only when the initial estimate is within the region of attraction of the optimum point.

Preprint submitted to 2009 IEEE International Conference on Robotics and Automation. Received September 15, 2008.

where $p(\mathbf{x}_0)$ is a prior on the robots' initial poses. Using the Markov and the Gaussian noise assumptions, the monotonicity of the logarithmic function, and the independence of the process and measurement noises, (3) simplifies into the following non-linear least-squares problem

$$\hat{\mathbf{x}}_{0:K-1} = \arg\min \left(\sum_{i=1}^{N} \sum_{\substack{j=1 \\ j\neq i}}^{N} \sum_{k=0}^{K-1} ||\mathbf{h}(\mathbf{x}_k^i, \mathbf{x}_k^j) - \mathbf{z}_k^{i,j}||^2_{\mathbf{R}_k^{i,j}} + \sum_{i=1}^{N} \sum_{k=0}^{K-2} ||\mathbf{f}(\mathbf{x}_k^i, \mathbf{u}_k^i, \mathbf{w}_k^i) - \mathbf{x}_{k+1}^i||^2_{\mathbf{Q}_k^i} + \sum_{i=1}^{N} ||\mathbf{x}_0^i - \mathbf{x}_{init}^i||^2_{\mathbf{P}_0^i} \right) \quad (4)$$

where $\mathbf{x}_{init}^i$ is the mean of the prior for the pose of robot i and $||\mathbf{e}||^2_{\mathbf{W}} = \mathbf{e}^T\mathbf{W}^{-1}\mathbf{e}$ is the weighted squared L_2-norm for a given covariance $\mathbf{W}$. Since the process (cf. (1)) and the measurement (cf. (2)) models are non-linear, the minimization problem in (4) is solved by iteratively linearizing about the latest estimates for the robots' poses. Each iteration of the non-linear minimization problem has the form

$$\begin{aligned} \delta\mathbf{x}^* = \arg\min & \left(\sum_{i=1}^{N} \sum_{k=0}^{K-2} ||(\overline{\mathbf{G}}_k^i)^{-1}\delta\mathbf{x}_{k+1}^i - \overline{\mathbf{F}}_k^i\delta\mathbf{x}_k^i - (\overline{\mathbf{G}}_k^i)^{-1}(\mathbf{f}(\hat{\mathbf{x}}_k^i, \mathbf{u}_k^i, 0) - \hat{\mathbf{x}}_{k+1}^i)||_2^2 \right. \\ & + \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j\neq i}}^{N} \sum_{k=0}^{K-1} ||(\mathbf{R}_k^{i,j})^{-1/2}(\mathbf{z}_k^{i,j} - \mathbf{h}(\hat{\mathbf{x}}_k^i, \hat{\mathbf{x}}_k^j)) - {}^i\overline{\mathbf{H}}_k^{i,j}\delta\mathbf{x}_k^i - {}^j\overline{\mathbf{H}}_k^{i,j}\delta\mathbf{x}_k^j||_2^2 \\ & \left. + \sum_{i=1}^{N} ||(\mathbf{P}_0^i)^{-1/2}\delta\mathbf{x}_0^i + (\mathbf{P}_0^i)^{-1/2}(\hat{\mathbf{x}}_0^i - \mathbf{x}_{init}^i)||_2^2 \right) \end{aligned} \quad (5)$$

where $\overline{\mathbf{F}}_k^i$, $\overline{\mathbf{G}}_k^i$, and ${}^i\overline{\mathbf{H}}_k^{i,j}$ are the pre-whitened Jacobians of the motion model with respect to the state and the odometry measurements, and of the measurement model with respect to the state. These are obtained by transforming the weighted squared L_2-norm into the regular squared L_2-norm as $||\mathbf{e}||^2_{\mathbf{W}} = (\mathbf{W}^{-1/2}\mathbf{e})^T(\mathbf{W}^{-1/2}\mathbf{e}) = ||\mathbf{W}^{-1/2}\mathbf{e}||_2^2$.

By stacking the different terms from (5) in a matrix $\mathbf{A}$ and a vector $\mathbf{b}$, the p^{th} iteration of the iterative minimization process is represented as

$$\delta\mathbf{x}^* = \arg\min ||\mathbf{A}(\hat{\mathbf{x}}_{0:K-1}^{(p)})\delta\mathbf{x} - \mathbf{b}(\hat{\mathbf{x}}_{0:K-1}^{(p)})||_2^2 \quad (6)$$

$$\hat{\mathbf{x}}_{0:K-1}^{(p+1)} = \hat{\mathbf{x}}_{0:K-1}^{(p)} + \delta\mathbf{x}^* \quad (7)$$

where $\mathbf{A}$ and $\mathbf{b}$ depend on the current iterate $\hat{\mathbf{x}}_{0:K-1}^{(p)}$. Details of this derivation and examples illustrating the structure of $\mathbf{A}$ and $\mathbf{b}$ are presented in [17].

IV. Centralized Cooperative Localization

We hereafter discuss the main drawbacks of a centralized approach to the minimization problem (5). In this case, all robots in the team periodically send their proprioceptive and exteroceptive measurements to a leader robot, or a Fusion Center (FC), that solves (5) and provides updated estimates for the robots' poses. Typically, algorithms such as the LM (cf. Alg. 1, [18]), that combines the Gauss-Newton and the Gradient Descent methods [18], are used.

The main drawback of a centralized approach is that it is susceptible to failures of the FC. Additionally, there is significant loss in terms of efficiency and speed due to the fact that a centralized algorithm does not utilize the available computation and storage resources in the robot team, i.e., while the FC is burdened with all the necessary computations, the other robots in the team remain idle after communicating their measurements. At each iteration of the LM algorithm, the FC must solve the modified normal equations

$$(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})\delta\mathbf{x} = \mathbf{A}^T\mathbf{b} \quad (8)$$

Since $\mathbf{H} = \mathbf{A}^T\mathbf{A}$ and $\mathbf{e} = \mathbf{A}^T\mathbf{b}$ have dimensions, $KN \times KN$ and $KN \times 1$ respectively, where K is the number of time-steps considered, this process requires $O(KN^3)$ operations [19, Section 4.3]. Moreover, once $\delta\mathbf{x}$ is computed, the FC must calculate the new estimates $(\hat{\mathbf{x}}_{0:K-1}^{(p)} + \delta\mathbf{x})$ and update $\mathbf{H}$ and $\mathbf{e}$. As K grows, the FC will have increasing difficulty not only in generating real-time solutions but also in handling the memory requirements for reevaluating and storing $\mathbf{A}$, $\mathbf{b}$ and other intermediate results.

In the next section, in order to address the limitations of the centralized approach, we present our distributed algorithm that leverages the memory and processing capabilities of all the robot team members and reduces the computational complexity of the CL problem.

Algorithm 1 LM Algorithm

Require: Initial guess $\hat{\mathbf{x}}_{0:K-1}^{(p)}$
Ensure: $\chi^2(\hat{\mathbf{x}}_{0:K-1}^{(p)} + \delta\mathbf{x}) - \chi^2(\hat{\mathbf{x}}_{0:K-1}^{(p)}) \leq 0.01\chi^2(\hat{\mathbf{x}}_{0:K-1}^{(p)})$ {Stopping Criterion}
1: Compute $\chi^2(\hat{\mathbf{x}}_{0:K-1}^{(p)}) = (\mathbf{A}^T\mathbf{b})^T\mathbf{A}^T\mathbf{b}$
2: Initialize $\lambda \leftarrow 0.001$ {Typical}
3: Solve $(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})\delta\mathbf{x} = \mathbf{A}^T\mathbf{b}$ and Evaluate $\chi^2(\hat{\mathbf{x}}_{0:K-1}^{(p)} + \delta\mathbf{x})$ {$\mathbf{I}$: Identity matrix}
4: **if** $\chi^2(\hat{\mathbf{x}}_{0:K-1}^{(p)} + \delta\mathbf{x}) \geq \chi^2(\hat{\mathbf{x}}_{0:K-1}^{(p)})$ **then**
5: $\quad\lambda \leftarrow \lambda \times 10$, Goto 3
6: **end if**
7: **if** $\chi^2(\hat{\mathbf{x}}_{0:K-1}^{(p)} + \delta\mathbf{x}) < \chi^2(\hat{\mathbf{x}}_{0:K-1}^{(p)})$ **then**
8: $\quad\lambda \leftarrow \lambda/10$, $\hat{\mathbf{x}}_{0:K-1}^{(p)} \leftarrow \hat{\mathbf{x}}_{0:K-1}^{(p)} + \delta\mathbf{x}$, Goto 3
9: **end if**
10: **return** $\hat{\mathbf{x}}_{0:K-1} \leftarrow \hat{\mathbf{x}}_{0:K-1}^{(p)}$

V. Distributed Cooperative Localization

A. Distributed Data Storage and Updating

In contrast to the centralized formulation that requires communication of all proprioceptive and exteroceptive measurements to the FC, in the proposed algorithm, each robot i constructs and updates rows/columns i, $i+N$, ... , $i+(K-1)N$ of $\mathbf{H}$ and the corresponding elements of $\mathbf{e}$. As an example,[2] in Fig. 1, robot 1 is responsible for rows/columns 1, 4 and 7, robot 2 for rows/columns 2, 5 and 8, and robot 3 for rows/columns 3, 6 and 9 of $\mathbf{H}$, and each of them is responsible for the corresponding elements of $\mathbf{e}$.

Consider the fifth row of $\mathbf{H}$ stored by robot 2 which contains the following 3 types of terms:[3] (i) Off-diagonal terms

[2]In order to simplify the presentation of the computation and communication complexity, we will consider the 1D case where each block of $\mathbf{H}$ (cf. Fig. 1) reduces to a scalar. Note that the ensuing analysis also holds for the case of robots navigating in 2D and 3D.

[3]For further details the interested reader is referred to [17].

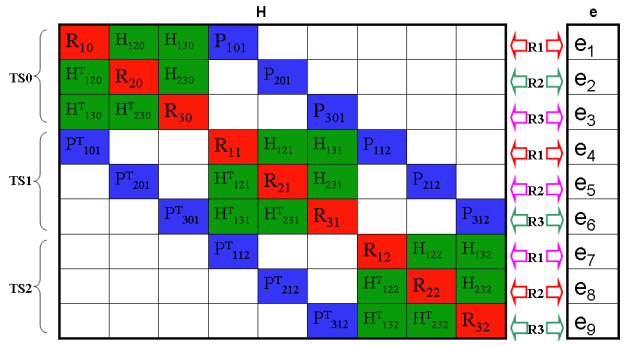

Fig. 1. Distribution of matrix **H** and vector **e** amongst the robots for an example of 3 robots navigating over 3 time-steps with a *complete* measurement graph (i.e., all robots measure the relative position of all other robots). TS: Time-step. H_{ijk} denotes off-diagonal terms corresponding to robot-to-robot measurements between robots i and j at time-step k. P_{ijk} denotes off-diagonal terms due to the motion model of robot i from time step j to time-step k. R_{ij} denotes the diagonal terms corresponding to both motion and robot-to-robot measurements involving robot i at time-step j.

(green), H_{231} and H_{121}, involving relative position measurement Jacobians evaluated at the robot pose estimates $(\hat{x}_1^2,\ \hat{x}_1^3)$ and $(\hat{x}_1^1,\ \hat{x}_1^2)$, respectively. (ii) Off-diagonal terms (blue), P_{201} and P_{212}, involving motion Jacobians between time-steps 0 and 1, and time-steps 1 and 2, evaluated at $\hat{x}_0^2$ and $\hat{x}_1^2$, respectively. (iii) Diagonal term (red), R_{21}, which contains information from the robot-to-robot measurements involving robot 2 at time-step 1 and from the motion model of robot 2 between time-steps 0 and 1, and time-steps 1 and 2. Additionally, computing the fifth element, e_5, of **e** requires estimates, $\hat{x}_0^2$, $\hat{x}_1^2$ and $\hat{x}_2^2$ of robot 2, estimates $\hat{x}_1^j$ of robot j ($j=1$ or 3) and the measurements, $z_1^{2,j}$ and $z_1^{j,2}$, between robots 2 and j.

While the estimates $\hat{x}_0^2$, $\hat{x}_1^2$ and $\hat{x}_2^2$ and measurements $z_1^{2,j}$ are locally available to robot 2, measurements $z_1^{j,2}$ and estimates $\hat{x}_1^j$ of robot j are necessary in order to construct the fifth row of **H** and e_5. These quantities can be easily obtained if at time-step 1, when robots 2 and j observe each other, robot j communicates its measurement and current state estimate to robot 2. By ensuring that these quantities are communicated by every robot when relative measurements are recorded, each robot can construct its assigned rows of **H** and elements of **e** with a minimal communication overhead of $O(N-1)$ per robot.

Note also, that by employing this distributed storage scheme, every time a new state estimate becomes available, the elements of the Hessian **H** and the residual **e** can be updated in parallel by the corresponding robots. Based on this distributed storing and updating approach, in the next section we present the distributed conjugate gradient algorithm for computing the updated state estimates during each iteration of the LM minimization.

B. Distributed Conjugate Gradient

As mentioned in Section IV, each iteration of the LM minimization process requires solving a system of normal equations (cf. (8)). Two types of algorithms can be used in this process: *direct* or *iterative* [20]. Direct algorithms, which have computational complexity of $O(KN^3)$ for banded systems [19], include methods such as Gauss-Elimination and its variants, Odd-Even Reduction, and Givens Rotations. Although the computational complexity of distributed implementations of these algorithms is $O(KN^2)$ [20], they have several disadvantages. The Odd-Even Reduction requires the inversion of $N\times N$ matrices at each time-step, making it numerically unstable, while Givens Rotations incur excessive communication overhead. Moreover, direct algorithms provide *no intermediate solution*. This is a major drawback especially when considering robots communicating via wireless connections susceptible to intermittent failures. If for any reason, the communication between the robots is interrupted before the direct algorithm has completed all its steps, the robots will have no new solution. In this case, they will have to revert to their previous estimates after having wasted valuable computation and communication resources.

In contrast, iterative algorithms, also referred to as *anytime* algorithms, generate an approximate solution at every iteration with increasing accuracy [20]. However, most of the commonly used iterative algorithms such as Jacobi, Gauss-Seidel, Jacobi overrelaxation, and Successive overrelaxation, converge only asymptotically (i.e., after infinite number of steps) [20]. Alternatively, the Conjugate Gradient (CG) algorithm is guaranteed to converge in at most KN iterations. Moreover, and for the special class of large systems of equations considered here, where **H** is a symmetric positive definite $KN\times KN$ matrix, the CG yields sufficiently accurate solutions with significantly fewer iterations [20].

In this section, we analyze the computational and communication complexity of the CG algorithm for *complete* measurement graphs, i.e., when each robot observes all other robots at every time step, leading to a total of $N(N-1)$ relative position measurements per time step. The details of a similar analysis for αN measurements per time-step, where $\alpha\in[1,\ (N-1)]$, can be found in [17]. Additionally, we compare and contrast the centralized CG (CCG), (single processor implementation) with the distributed CG (DCG) (multi-processor implementation). Table I lists the steps required during each iteration of the CG along with their computational and communication complexity. Specifically, each iteration m, where $m\in\{0,\ldots,KN-1\}$, of the CG consists of the following steps:

Step 1: $\mathbf{g}_m=\mathbf{H}\delta\mathbf{x}_m-\mathbf{e}$

Here $\mathbf{g}_m=[g_m(1)\ldots g_m(KN)]^T$ is a $KN\times 1$ vector. All robots initialize $\delta\mathbf{x}_0$ to a vector of zeros.

Due to the special block tri-diagonal structure of **H**, computing each element $g_m(j)$ of $\mathbf{g}_m$, where $j=1,\ldots,KN$, requires $O(N)$ operations.

1) **CCG**: Calculating KN elements of $\mathbf{g}_m$ requires $O(KN^2)$ operations.
2) **DCG**: Given the distribution of the rows of **H** and **e** among the robots (cf. Section V-A), robot i calculates $g_m(j)$, where $j\in S_i=\{i,\ i+N,\ i+2N,\ldots,\ i+(K-1)N\}$, locally, i.e., each robot calculates K terms of $\mathbf{g}_m$, requiring $O(KN)$ operations per robot (cf. Fig. 2).

For this step, the communication cost is zero, as all computations are carried out locally by the robots.

Step 2: $\beta_m=\mathbf{g}_m{}^T\mathbf{g}_m/\mathbf{g}_{m-1}{}^T\mathbf{g}_{m-1}$

For initialization, $\beta_0=0$. Also, $\mathbf{g}_m$ is generally a dense

TABLE I

COMPLEXITY ANALYSIS OF THE CONJUGATE GRADIENT METHOD

Algorithm		Computation				Communication	
		Centralized		Distributed		Distributed	
Number of Measurements		$N(N-1)$	αN	$N(N-1)$	αN	$N(N-1)$	αN
Step 1	$\mathbf{g}_m = H\delta\mathbf{x}_m - e$	$O(KN^2)$	$O(\alpha KN)$	$O(KN)$	$O(\alpha K)$	0	0
Step 2	$\beta_m = \mathbf{g}_m{}^T\mathbf{g}_m / \mathbf{g}_{m-1}{}^T\mathbf{g}_{m-1}$	$O(KN)$	$O(KN)$	$O(K+log(N))$	$O(K+log(N))$	$O(1)$	$O(1)$
Step 3	$\mathbf{s}_m = -\mathbf{g}_m + \beta_m\mathbf{s}_{m-1}$	$O(KN)$	$O(KN)$	$O(K)$	$O(K)$	0	0
Step 4a	$\mathbf{s}_m{}^T\mathbf{g}_m$	$O(KN)$	$O(KN)$	$O(K+log(N))$	$O(K+log(N))$	O(1)	O(1)
Step 4b	$\mathbf{h}_m = H\mathbf{s}_m$	$O(KN^2)$	$O(\alpha KN)$	$O(KN)$	$O(\alpha K)$	$O(K)$	$O(K)$
Step 4c	$\gamma_m = -\mathbf{s}_m{}^T\mathbf{g}_m / \mathbf{s}_m^T\mathbf{h}_m$	$O(KN)$	$O(KN)$	$O(K+log(N))$	$O(K+log(N))$	$O(1)$	$O(1)$
Step 5	$\delta\mathbf{x}_{m+1} = \delta\mathbf{x}_m + \gamma_m\mathbf{s}_m$	$O(KN)$	$O(KN)$	$O(KN)$	$O(KN)$	0	0

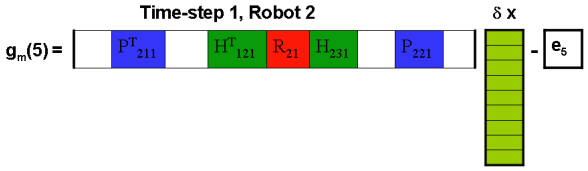

Fig. 2. Example of Robot 2 calculating element $g_m(5)$ of $\mathbf{g}_m$.

$KN \times 1$ vector.

1) **CCG:** The computational complexity of computing the inner-product, $\mathbf{g}_m{}^T\mathbf{g}_m$, is $O(KN)$.
2) **DCG:** Each robot i calculates the dot-product $b_i = \sum_{j\in S_i} g_m(j)^2$, $i = 1,\ldots,N$, of its K local elements of $\mathbf{g}_m$ at a cost of $O(K)$. As is well known, adding these N scalars in a distributed way requires $O(\log N)$ steps[4] with communication cost of O(1) per robot [20, Sec.1.2.3]. At the end of this process, one of the robots acquires the final result for $\mathbf{g}_m{}^T\mathbf{g}_m$ and calculates β_m using the value of $\mathbf{g}_{m-1}{}^T\mathbf{g}_{m-1}$ from the previous iteration, at a cost of $O(1)$. Once β_m is available, it is broadcasted to all other robots at a communication cost of $O(1)$. Thus, for this step, the computation and communication cost per robot is $O(K + logN)$ and $O(1)$, respectively.

Step 3: $\mathbf{s}_m = -\mathbf{g}_m + \beta_m\mathbf{s}_{m-1}$

For $m = 1$, $\mathbf{s}_0$ is initialized to $\mathbf{g}_0$. This step incurs no communication overhead since all computations are local.

1) **CCG:** Since $\mathbf{g}_m$ and $\mathbf{s}_{m-1}$ are vectors of dimension $KN \times 1$ and β_m is a scalar, the computational complexity for calculating $\mathbf{s}_m$ is $O(KN)$.
2) **DCG:** Due to the distribution of $\mathbf{g}_m$ (cf. Step 1), robot i calculates $s_m(j)$, where $j \in S_i$, locally, i.e., each robot evaluates K terms of $\mathbf{s}_m$, which requires $O(K)$ operations per robot.

Step 4: $\gamma_m = -\mathbf{s}_m{}^T\mathbf{g}_m / \mathbf{s}_m{}^T\mathbf{H}\mathbf{s}_m$

We analyze the cost of calculating $\mathbf{s}_m{}^T\mathbf{g}_m$ and $d_m = \mathbf{s}_m{}^T\mathbf{H}\mathbf{s}_m$ separately.

- **Calculate** $\mathbf{s}_m{}^T\mathbf{g}_m$:
  1) **CCG**: The computational complexity is $O(KN)$.
  2) **DCG**: As in Step 2, robot i has K elements each of $s_m(j)$ and $g_m(j)$, $j \in S_i$, locally available. The partial dot-product $\sum_{j\in S_i} = s_m(j)g_m(j)$ computed by robot i requires $O(K)$ operations, resulting into N scalars. The rest of the analysis is identical to Step 2.
- **Calculate** d_m:
  1) **CCG**: Similar to Step 1, calculating $\mathbf{h}_m = \mathbf{H}\mathbf{s}_m$ requires $O(KN^2)$ operations, while computing $\mathbf{s}_m{}^T\mathbf{h}_m$ has cost $O(KN)$.
  2) **DCG**: For calculating $\mathbf{h}_m$, all robots must acquire $\mathbf{s}_m$. Thus, each robot broadcasts its K elements of $\mathbf{s}_m$, with total communication cost of $O(KN)$, or $O(K)$ per robot. Subsequently, each robot i calculates $h_m(j)$, $j \in S_i$, locally with cost $O(KN)$ per robot (cf. Step 1). Computation of the dot-product $\mathbf{s}_m{}^T\mathbf{h}_m$ is similar to Step 2, and has computation and communication cost of $O(K+\log N)$ and $O(1)$, respectively.

  Once d_m and $\mathbf{s}_m{}^T\mathbf{g}_m$ are available, γ_m is calculated for the computational cost of $O(1)$ and broadcasted.

Step 5: $\delta\mathbf{x}_{m+1} = \delta\mathbf{x}_m + \gamma_m\mathbf{s}_m$

Since γ_m, $\mathbf{s}_m$ and $\delta\mathbf{x}_m$ are locally available, each robot calculates $\delta\mathbf{x}_{m+1}$ at the computational cost of $O(KN)$.

Steps 1 to 5 are repeated until convergence, i.e. $\mathbf{g}_m = \mathbf{0}$. For each iteration of the CCG, the computational complexity is $O(KN^2)$, while for the DCG it is $O(KN)$. Theoretically, KN such iterations are necessary, which makes the complexity of the CCG $O(K^2N^3)$, while $O(K^2N^2)$ operations per robot are required by the DCG, thus reducing the computational complexity of CL by a factor of N. Moreover, since $\mathbf{H}$ and $\mathbf{e}$ are stored distributively, the time required for updating them is also reduced by a factor of N. Additionally, since processing is distributed, the system is more robust to failures. If robot i fails, the team simply discards the rows, columns of $\mathbf{H}$ and elements of $\mathbf{e}$ corresponding to robot i and carries out CL on the remaining data.

Given the solution $\delta\mathbf{x}$ from the DCG, the robots compute the new estimates for $\mathbf{x}$ and update $\mathbf{H}$ and $\mathbf{e}$. This constitutes a single iteration of the LM algorithm (cf. Step 3 of Alg. 1). Note that the performance of the DCG, is identical to that of the centralized MAP-based CL since *no* approximations have been introduced in the distributed algorithm.

A limitation of the MAP-based CL is that as the number of time steps, K, increases, so does the computational and storage requirement. Typically, this problem is addressed by marginalizing past robot poses and maintaining a constant length time window. In the next section, we demonstrate how marginalization is efficiently implemented in our distributed framework at reduced computation cost.

[4]Note that the computation cost per robot is constant, $O(1)$. However, we are primarily interested in the time required for performing these additions and thus we adopt $O(\log N)$ as the computation cost of these operations.

Preprint submitted to 2009 IEEE International Conference on Robotics and Automation. Received September 15, 2008.

C. Marginalization of Past Robot Poses

In this section, we first discuss marginalization in the context of CL and then present its distributed implementation along with the complexity analysis. Depending on the computational and communication resources, we restrict the dimension of the minimization problem to J. Therefore, when $KN = J$, we need to marginalize at least the robot poses from time-step 0 in order to reduce the size of the problem to $(K-1)N$. This ensures that there will be sufficient resources for processing the measurements corresponding to the next time step.

Consider the case of 3 robots over 4 time steps ($k = 0,\ldots,3$) with a complete measurement graph and let $J = 12$. In order to process measurements from time-step 4, we need to marginalize the robot poses from time-step 0. The corresponding cost function, linearized about the current estimates, $\hat{\mathbf{x}}_{0:3}$, is

$$\begin{aligned}\eta = &\sum_{i=1}^{3}\sum_{k=0}^{2}||(\overline{\mathbf{G}}_k^i)^{-1}\delta\mathbf{x}_{k+1}^i - \overline{\mathbf{F}}_k^i\delta\mathbf{x}_k^i - (\overline{\mathbf{G}}_k^i)^{-1}(\mathbf{f}(\hat{\mathbf{x}}_k^i,\mathbf{u}_k^i,0) - \hat{\mathbf{x}}_{k+1}^i)||_2^2 \\ &+\sum_{i=1}^{3}\sum_{\substack{j=1\\ j\neq i}}^{3}\sum_{k=0}^{3}||(\mathbf{R}_k^{i,j})^{-1/2}(\mathbf{z}_k^{i,j} - \mathbf{h}(\hat{\mathbf{x}}_k^i,\hat{\mathbf{x}}_k^i)) - {}^i\overline{\mathbf{H}}_k^{i,j}\delta\mathbf{x}_k^i - {}^j\overline{\mathbf{H}}_k^{i,j}\delta\mathbf{x}_k^j||_2^2 \\ &+\sum_{i=1}^{3}||(\mathbf{P}_0^i)^{-1/2}\delta\mathbf{x}_0^i + (\mathbf{P}_0^i)^{-1/2}(\hat{\mathbf{x}}_0^i - \mathbf{x}_{init}^i)||_2^2 \end{aligned} \quad (9)$$

The Hessian matrix $\mathbf{H}$ and vector $\mathbf{e}$ in the normal equations corresponding to (9) can be split into sub-matrices and sub-vectors as shown in Fig. 3. Here $\mathbf{A}$ and $\mathbf{B}$ are $N\times N$ diagonal and $N\times(K-1)N$ off-diagonal blocks of $\mathbf{H}$ respectively, that depend on $\mathbf{x}_0$. Note that $\mathbf{B}$ is a sparse matrix with only N elements along the diagonal. This $N\times N$ non-zero sub-matrix of $\mathbf{B}$ is denoted as $\mathbf{B}_{trunc}$. Also, $\mathbf{c}_m$ is a $N\times 1$ vector of $\mathbf{e}$ that depends on $\mathbf{x}_0$.

Before proceeding with the marginalization process, we examine the structure of the elements of $\mathbf{H}$ and $\mathbf{e}$ that will be affected. Specifically, the block diagonal $N\times N$ sub-matrix, $\mathbf{D}_1 = \mathbf{D}_{1c} + \mathbf{D}_{1m}$, of $\mathbf{H}$ comprises of $\mathbf{D}_{1c}$ that depends on $\mathbf{x}_1$ and will continue to be updated as new estimates are computed, and $\mathbf{D}_{1m}$, which depends on $\mathbf{x}_0$ and will remain constant after the marginalization. Similarly, $\mathbf{c}_1 = \mathbf{c}_{1c} + \mathbf{c}_{1m}$, where $\mathbf{c}_{1c}$ depends on $\mathbf{x}_1$ and $\mathbf{c}_{1m}$ depends on $\mathbf{x}_0$.

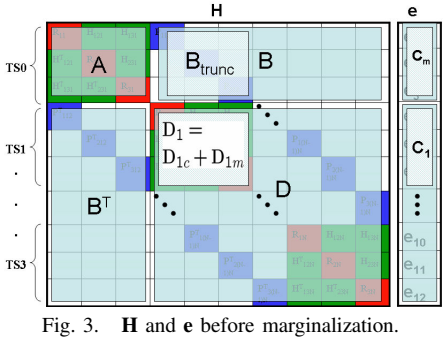

Fig. 3. $\mathbf{H}$ and $\mathbf{e}$ before marginalization.

Marginalization of robot poses from time-step 0 (i.e., $\mathbf{x}_0$) requires that we fix a value for the corresponding variable $\delta\mathbf{x}_0$ in (9), treat it as a constant, and do not estimate it in the future. This value is determined by differentiating the linearized cost function with respect to $\delta\mathbf{x}_0$ and setting it equal to zero, which gives us:

$$\delta\mathbf{x}_0 = \mathbf{A}^{-1}\mathbf{c}_m - \mathbf{A}^{-1}\mathbf{B}_{trunc}\delta\mathbf{x}_1 \quad (10)$$

Substituting $\delta\mathbf{x}_0$ in the linearized cost function (9) yields the marginalized cost function, which will no longer contains terms involving $\delta\mathbf{x}_0$ [17].

The system of normal equations corresponding to the marginalized cost function is $\mathbf{H}_{new}[\delta\mathbf{x}_1\ldots\delta\mathbf{x}_3]^T = \mathbf{e}_{new}$. The structure of $\mathbf{H}_{new}$ and $\mathbf{e}_{new}$ is shown in Fig. 4. Note that as a result of marginalization, the correlation between robot poses $\mathbf{x}_0$ and $\mathbf{x}_1$ (due to propagation) introduces additional terms in the new $\mathbf{D}_1$ and $\mathbf{c}_1$, denoted as [17]:

$$\mathbf{D}_1^* = \mathbf{D}_{1c} + \mathbf{D}_{mod} \quad , \quad \mathbf{D}_{mod} = \mathbf{D}_{1m} - \mathbf{B}^T\mathbf{A}\mathbf{B}$$
$$\mathbf{c}_1^* = \mathbf{c}_{1c} - \mathbf{D}_{mod}\left(\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_{1_{margin}}\right) + \mathbf{c}_{mod},\ \mathbf{c}_{mod} = \mathbf{c}_{1m} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{c}_m$$

In the above expressions $\mathbf{D}_{mod}$ remains constant and needs to be stored and added to the new Hessian matrix $\mathbf{H}_{new}$ at every iteration of the minimization algorithm (note that $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{D}_{1m}$ involve Jacobians evaluated at the estimate of $\mathbf{x}_0$ at the time of the marginalization, denoted by $\hat{\mathbf{x}}_{0_{margin}}$; in contrast, $\mathbf{D}_{1c}$, consists of Jacobians evaluated at the estimate of $\mathbf{x}_1$, which changes as new measurements are obtained). Similarly, out of the terms comprising $\mathbf{c}_1^*$, only $\mathbf{c}_{1c}$ that depends on the estimate of $\mathbf{x}_1$ will be updated in future time steps (note that $\mathbf{c}_{1m}$ and $\mathbf{c}_m$ are evaluated at $\hat{\mathbf{x}}_{0_{margin}}$ and $\hat{\mathbf{x}}_{1_{margin}}$. Here, $\hat{\mathbf{x}}_{1_{margin}}$ denotes the stored estimate for $\mathbf{x}_1$ at the time of the marginalization).

In summary, after each marginalization three terms have to be stored: (i) $\mathbf{D}_{mod}$, (ii) $\mathbf{c}_{mod}$, and (iii) $\hat{\mathbf{x}}_{1_{margin}}$. Note that the dimensions of the above 3 quantities remain the same, irrespective of the number of time steps being marginalized simultaneously. If the first p time steps are marginalized, only $\hat{\mathbf{x}}_{p+1_{margin}}$, of dimension $N\times 1$ will have to be stored as a result of the correlations between time-steps p and $p+1$. Also, the dimensions of $\mathbf{D}_{mod}$ and $\mathbf{c}_{mod}$ remain $N\times N$ and $N\times 1$, respectively.

The 2nd column of Table II lists the steps involved in marginalization. Step 1 requires the inversion of the $lN\times lN$ matrix $\mathbf{A}$, where l is the number of time-steps being marginalized. Thus, when the number of robots N and/or steps l is large, computation of this dense inverse can be a bottleneck ($O(l^3N^3)$) [19].

To address this problem, we use the distributed Gauss-Jordan method [20]. Instead of inverting $\mathbf{A}$ separately, we

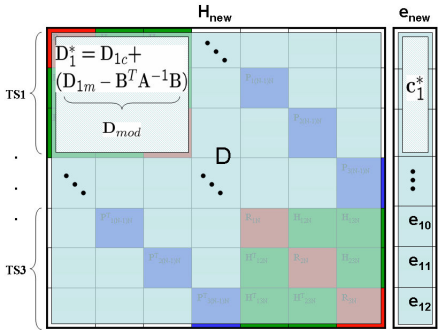

Fig. 4. $\mathbf{H}_{new}$ and $\mathbf{e}_{new}$ after marginalization.

TABLE II
COMPLEXITY ANALYSIS OF MARGINALIZATION ($l = 1$)

		Computation		Comm.
Algorithm		Centralized	Distributed	Distributed
Step 1	$\mathbf{A}^{-1}\mathbf{B}$ and $\mathbf{A}^{-1}\mathbf{c}_m$	$O(N^3)$	$O(N^2)$	$O(N)$
Step 2a	$\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}$	$O(N^2)$	$O(N)$	0
Step 2b	$\mathbf{B}^T\mathbf{A}^{-1}\mathbf{c}_m$	$O(N)$	$O(1)$	0
Step 3a	$\mathbf{D}_{1m} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}$	$O(N)$	$O(1)$	0
Step 3b	$\mathbf{c}_{1m} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{c}_m$	$O(N)$	$O(1)$	0

Algorithm 2 Distributed Gauss-Jordan

for $i = 1$ to $N-1$ **do**
- Divide Row i by a_{ii}

Computation cost $= N+2$ $\{N-i+1$ elements in the Row i of matrix $\mathbf{A}$, i elements in Row i of matrix $\mathbf{B}_{trunc}$ and i^{th} element of $\mathbf{c}_m\}$
Communication cost $= 0$
- Broadcast Row i of robot i to all other robots

Computation cost $= 0$
Communication cost $= N+1$ $\{N-i$ elements in Row i of matrix $\mathbf{A}$ as all elements until a_{ii} do not need to be communicated, i elements in Row i of matrix $\mathbf{B}_{trunc}$ and the i^{th} element of $\mathbf{c}_m\}$
for $j = 1$ to N ; $j \neq i$ **do**
- Compute $Row_j = Row_j - pivot \times Row_i$ {No need to compute the pivot as the pivot element will be the same as a_{ji}}

Computation cost $= 2N+3$.
Communication cost $= 0$.
end for{This operation is simultaneously carried out by all j robots}
end for

calculate the quantities $\mathbf{A}^{-1}\mathbf{B} = \left[\mathbf{A}^{-1}\mathbf{B}_{trunc}\ \mathbf{0}\right]$ and $\mathbf{A}^{-1}\mathbf{c}_m$ directly. Specifically, a new augmented matrix $\mathbf{M} = [\mathbf{A}\ \mathbf{B}\ \mathbf{c}_m]$ is considered and using the Gauss-Jordan algorithm, it is reduced to $[\mathbf{I}\ \mathbf{B}^*\ \mathbf{c}_m^*]$, where $\mathbf{I}$ is the identity matrix of the same dimensions as $\mathbf{A}$. The resulting terms $\mathbf{B}^*$ and $\mathbf{c}_m^*$ are equal to $\mathbf{A}^{-1}\mathbf{B}$ and $\mathbf{A}^{-1}\mathbf{c}_m$, respectively. Algorithm 2 presents the distributed Gauss-Jordan method (for $l = 1$) which requires $O(N^2)$ operations and has communication cost $O(N)$ per robot. Here, it is interesting to note that Gauss-Jordan for the positive definite matrix $\mathbf{A}$ is numerically stable and hence does not require pivoting [19]. This further reduces its communication requirements.

Once Step 1 of the marginalization process is complete (cf. Table II), each robot has a row of $\mathbf{A}^{-1}\mathbf{B}$ and an element of $\mathbf{A}^{-1}\mathbf{c}_m$ stored locally. In Steps 2 and 3, each robot calculates a row of $\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}$ and then $\mathbf{D}_{mod} = \mathbf{D}_{1m} - \mathbf{B}^T\mathbf{A}\mathbf{B}$ (note that $\mathbf{D}_{1m}$ is diagonal), and an element of $\mathbf{B}^T\mathbf{A}^{-1}\mathbf{c}_m$, followed by $\mathbf{c}_{mod} = \mathbf{c}_{1m} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{c}_m$ locally with computation cost of $O(N)$ per robot. Thus, the computational complexity of the distributed implementation of marginalization is reduced by an order of magnitude to $O(N^2)$ (or $O(l^2N^2)$ for $l > 1$).

VI. SIMULATION RESULTS

The performance of the proposed distributed MAP-based localization algorithm was tested in simulation. We consider a team of 18 robots moving in 2D following phase-shifted sinusoidal trajectories. The robots move in an area of approximately 25 m $\times$ 90 m for 450 time steps (each time step has duration 0.05 sec). Each robot measures its linear, v, and rotational, ω, velocity, as well as its distance d and bearing θ to all other robots in the team. The noise in all measurements is modeled as zero-mean, white Gaussian and has standard deviation: $\sigma_v = 2\%v$, $\sigma_\omega = 1$ deg/sec for the linear and rotational velocity measurements, respectively, and $\sigma_d = 2\%d$ and $\sigma_\theta = 1$ deg for the corresponding distance and bearing measurements.

The minimization problem is solved every 5 time steps, over a sliding time window of $K = 10$ time steps, while marginalization is carried out every 5 time steps (i.e., the number of time steps considered in the estimated state vector varies between 5 and 10). We compare the performances of the following approaches for CL:

1) Centralized[5] EKF (computational complexity $O(N^4)$).
2) Distributed MAP-based estimator using the DCG algorithm and marginalization (computational complexity $O(K^2N^2)$).
3) Distributed MAP-based estimator using an approximate DCG algorithm and marginalization (computational complexity $O(KN^2)$). In this case, we allow the DCG algorithm to perform N iterations only.

As compared to the centralized EKF (approach 1), our proposed algorithm (approach 2) has reduced computational complexity when a small (compared to the size of the team) number of time steps K is considered, i.e., when $K \ll N$, $O(K^2N^2) \ll O(N^4)$. However, when the robots need to consider a large number of time steps, in order to reduce the effect of linearization errors, the approximate version of the DCG algorithm is used (approach 3). In this case, the DCG is allowed to run for N iterations only, for a total cost of $O(KN^2)$. In general, the number of time steps K considered and the number of DCG iterations are design parameters that can be adjusted so as to trade processing for increased accuracy.

We employ the RMS error criterion to test the accuracy of these 3 approaches. Fig. 5 shows the RMS error in the robots' position estimates for every 20 time-steps for clarity (averaged over 30 runs). As evident, the distributed MAP-based estimator (approach 1) outperforms the centralized EKF in terms of accuracy. This is due to the fact that the MAP estimator reduces the linearization errors over all K time steps considered (sliding window smoothing), and thus improves the accuracy of the robots' pose estimates. Furthermore, we see that approach 3, which is an approximation of the distributed MAP-based estimator, is also more accurate than the EKF-based approach. This can be attributed to the fast convergence of the DCG for positive definite matrices (in this case the Hessian $\mathbf{H}$), resulting in accuracy comparable to approach 2, even when the DCG is allowed to run for only N iterations.

Fig. 6, that depicts the RMS error in the robots' orientation estimates at every 10 time-steps, corroborates the results of Fig 5. In this figure, the improvement in the accuracy of the robots' orientation estimates for the MAP-based algorithms,

[5]While we are not aware of any existing distributed implementation of EKF-based CL, we believe that by distributing the processing among N robots, this cost may be reduced from $O(N^4)$ to $O(N^3)$.

as compared to the centralized EKF, is more pronounced. Approach 2 is the most accurate, followed closely by approach 3, whose performance is almost indistinguishable from that of approach 2. As evident, both MAP estimators outperform the EKF in terms of accuracy, while they require fewer operations.

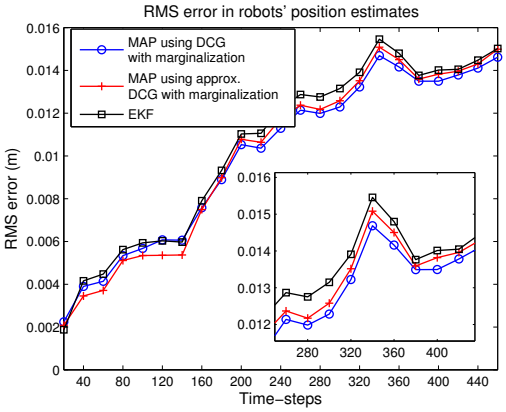

Fig. 5. RMS error in the robots' position estimates

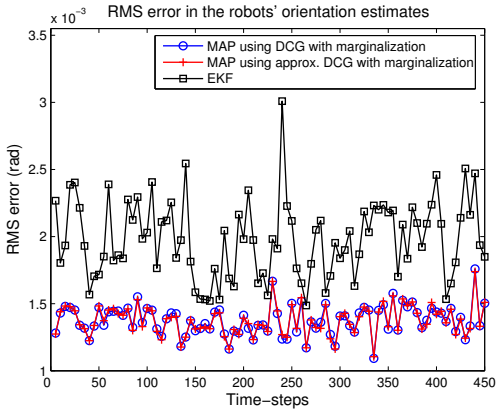

Fig. 6. RMS error in the robots' orientation estimates

VII. Conclusions and Future Work

In this paper, we introduce a novel distributed algorithm for MAP-based cooperative localization (CL) that utilizes all the available computational resources of a robot team to achieve real-time operation. The proposed algorithm, uses distributed data storage, the distributed conjugate gradient (DCG) algorithm, and distributed marginalization of past robot poses in order to spread the computations amongst the robots, and hence reduce the overall computational complexity of CL. Additionally, we have shown that by limiting the number of iterations of the DCG algorithm, the resulting approximate MAP estimator has accuracy almost indistinguishable of that of the MAP algorithm using the exact DCG, while significantly reducing the required number of operations. Simulation results demonstrate that the MAP-based CL algorithms (using the exact and the approximate DCG) outperform the Extended Kalman filter in terms of accuracy while having lower computational requirements.

A limitation of DCG is that it requires synchronous communication amongst the robots. This, however, may not be possible when robot teams operate under extreme environments with frequent communication failures. In order to address this issue, we will direct our future work towards developing algorithms that can tolerate asynchronous communication.

References

[1] T. L. Huntsberger, A. Trebi-Ollennu, H. Aghazarian, P. S. Schenker, P. Pirjanian, and H. D. Nayar, "Distributed control of multi-robot systems engaged in tightly coupled tasks," *Autonomous Robots*, vol. 17, no. 1, pp. 79–92, 2004.

[2] A. D. Tews, G. S. Sukhatme, and M. J. Mataric, "A multi-robot approach to stealthy navigation in the presence of an observer," in *Proc. of the IEEE International Conference on Robotics and Automation*, New Orleans, LA, Apr. 26–May 1, 2004, pp. 2379–2385.

[3] Y. Feng, Z. Zhu, and J. Xiao, "Heterogeneous multi-robot localization in unknown 3D space," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, Oct. 9-15, 2006, pp. 4533–4538.

[4] S. Roumeliotis and G. Bekey, "Distributed multirobot localization," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 781–795, Oct 2002.

[5] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Localization for mobile robot teams using maximum likelihood estimation," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, EPFL, Switzerland, Sept. 30–Oct. 5, 2002, pp. 434–439.

[6] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, no. 3, pp. 325–344, 2000.

[7] R. Kurazume, S. Nagata, and S. Hirose, "Cooperative positioning with multiple robots," in *Proc. of the IEEE International Conference on Robotics and Automation*, San Diego, CA, May 8-13, 1994, pp. 1250–1257.

[8] R. Grabowski, L. E. Navarro-Serment, C. J. J. Paredis, and P. K. Khosla, "Heterogeneous teams of modular robots for mapping and exploration," *Autonomous Robots*, vol. 8, no. 3, pp. 293–308, 2000.

[9] I. M. Rekleitis, G. Dudek, and E. E. Milios, "Multi-robot collaboration for robust exploration," in *Proc. of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, Apr. 24-28, 2000, pp. 3164–3169.

[10] S. I. Roumeliotis, "Robust mobile robot localization: from single-robot uncertainties to multi-robot interdependencies," Ph.D. dissertation, Los Angeles, CA, 2000.

[11] F. Dellaert, F. Alegre, and E. B. Martinson, "Intrinsic localization and mapping with 2 applications: Diffusion mapping and marco polo localization," in *Proc. of the IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, Sept. 14-19, 2003, pp. 2344–2349.

[12] S. Panzieri, F. Pascucci, and R. Setola, "Multirobot localization using interlaced extended kalman filter," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, Oct. 9-15, 2006, pp. 2816–2821.

[13] L. Glielmo, R. Setola, and F. Vasca, "An interlaced extended kalman filter," *IEEE Transactions on Automatic Control*, vol. 44, no. 8, pp. 1546–1549, Aug. 1999.

[14] N. Karam, F. Chausse, R. Aufrere, and R. Chapuis, "Localization of a group of communicating vehicles by state exchange," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, Oct. 9-15, 2006, pp. 519–524.

[15] A. Martinelli, "Improving the precision on multi robot localization by using a series of filters hierarchically distributed," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, Oct. 9–Nov. 2, 2007, pp. 1053–1058.

[16] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Localization for mobile robot teams: A distributed MLE approach," in *Experimental Robotics VIII*. Springer-Verlag, 2003, pp. 146–155.

[17] E. D. Nerurkar and S. I. Roumeliotis, "Distributed MAP estimation algorithm for cooperative localization," Dept. of Comp. Sci. & Eng., University of Minnesota, Tech. Rep., 2008. [Online]. Available: http://www-users.cs.umn.edu/∼nerurkar/Nerurkar_DstrbMAP.pdf

[18] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in C : the art of scientific computing*. Cambridge University Press, 1992.

[19] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Johns Hopkins University Press, 1983.

[20] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., 1989.